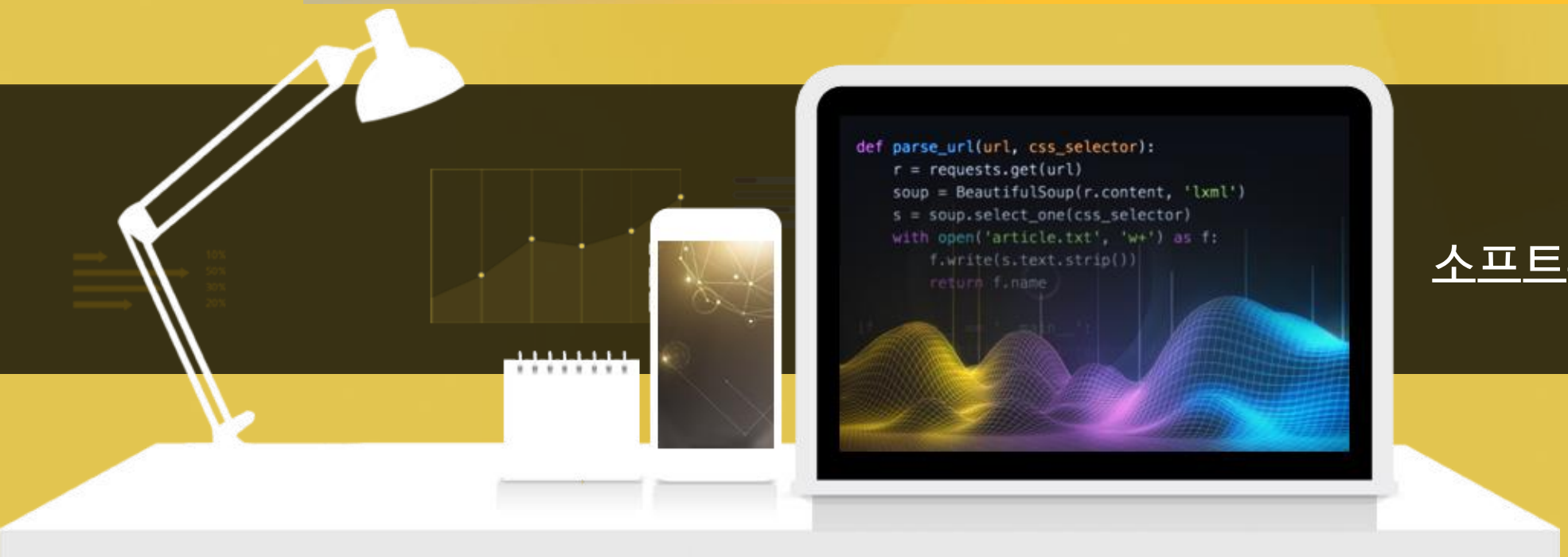
# 데이터 전처리




소프트웨어융합대학원
진혜진

# 01 데이터 전처리의 기초

## 1. 데이터 전처리의 개념

- 데이터 전처리(data preprocessing) : 머신러닝 모델에 훈련 데이터를 입력하기 전에 데이터를 가공
- 넘파이나 판다스 같은 머신러닝의 핵심 도구,
  맷플롯립과 시본 같은 데이터 시각화 도구를 활용하여 실제 데이터를 정리
- 머신러닝 기초 수식

$$y = f(X)$$

  - 데이터 X를 머신러닝 함수 f( )에 넣으면 그 결과 y가 나옴
  - 데이터 X는 훈련 데이터(train data)와 테스트 데이터(test data)가 모두 같은 구조를 갖는 피쳐(feature)이어야 함

## 2. 데이터 품질 문제

### 2.1 데이터 분포의 지나친 차이

- 데이터가 연속형 값인데 최댓값과 최솟값 차이가 피쳐보다 더 많이 나는 경우
- 학습에 영향을 줄 수 있기 때문에 데이터의 스케일(scale)을 맞춰줌
  - 데이터의 최댓값과 최솟값을 0에서 1 사이 값으로 바꾸거나 표준 정규분포 형태로 나타내는 등

# 01 데이터 전처리의 기초

## 2.2 기수형 데이터와 서수형 데이터

- 기수형 데이터와 서수형 데이터는 일반적으로 숫자로 표현되지 않음
- 컴퓨터가 이해할 수 있는 숫자 형태의 정보로 변형

## 2.3 결측치

- 결측치(missing data) : 실제로 존재하지만 데이터베이스 등에 기록되지 않는 데이터
- 해당 데이터를 빼고 모델을 돌릴 수 없기 때문에 결측치 처리 전략을 세워 데이터를 채워 넣음

# 01 데이터 전처리의 기초

## 2.4 이상치

- 이상치(outlier) : 극단적으로 크거나 작은 값
- 단순히 데이터 분포의 차이와는 다름
- 데이터 오기입이나 특이 현상 때문에 나타남

# 02 데이터 전처리의 전략

## 1. 결측치 처리하기 : 드롭과 채우기

- 데이터를 삭제하거나 데이터를 채움
  - 데이터가 없으면 해당 행이나 열을 삭제
  - 평균값, 최빈값, 중간값 등으로 데이터를 채움

# 02 데이터 전처리의 전략

In [1]:

```
import pandas as pd
import numpy as np

raw_data = {'first_name': ['Jason', np.nan, 'Tina', 'Jake', 'Amy'],
            'last_name': ['Miller', np.nan, 'Ali', 'Milner', 'Cooze'],
            'age': [42, np.nan, 36, 24, 73],
            'sex': ['m', np.nan, 'f', 'm', 'f'],
            'preTestScore': [4, np.nan, np.nan, 2, 3],
            'postTestScore': [25, np.nan, np.nan, 62, 70]}

df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age', 'sex',
'preTestScore', 'postTestScore'])
df
```

Out [1]:

| | first_name | last_name | age | sex | preTestScore | postTestScore |
|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 |
| 1 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | Tina | Ali | 36.0 | f | NaN | NaN |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 |

# 02 데이터 전처리의 전략

- 결측치를 확인할 때 isnull 함수 사용
  - NaN 값이 존재할 경우 True, 그렇지 않을 경우 False 출력

| In [2]: | df.isnull().sum() / len(df) |
|---|---|
| Out [2]: | first_name 0.2<br>last_name 0.2<br>age 0.2<br>sex 0.2<br>preTestScore 0.4<br>postTestScore 0.4<br>dtype: float64 |

- sum 함수로 True인 경우 모두 더하고 전체 데이터 개수로 나누어 열별 데이터 결측치 비율을 구함

# 02 데이터 전처리의 전략

## 1.1 드롭

- 드롭(drop) : 결측치가 나온 열이나 행을 삭제
- dropna 사용하여 NaN이 있는 모든 데이터의 행을 없앰

In [3]:
```
df.dropna()
```

Out [3]:

| | first_name | last_name | age | sex | preTestScore | postTestScore |
|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 |

# 02 데이터 전처리의 전략

## 드롭의 결과물 저장

- 드롭과 관련된 대부분의 명령어들은 실제 드롭한 결과를 반환하나 객체에 드롭 결과를 저장하지는 않음
- 드롭의 결과물을 저장하려면 다른 변수에 재할당
- 또는 매개변수 inplace=True 사용
  - 자체적으로 값이 변하면 이후에 해당 데이터를 불러 쓰거나 다시 코드를 실행할 때 문제가 되기 때문에 새로운 값에 복사하는 것이 좋음

| | |
|---|---|
| In [4]: | df_no_missing = df.dropna()<br>df_no_missing |

# 02 데이터 전처리의 전략

- 매개변수 how로 조건에 따라 결측치를 지움
  - how에는 매개변수 'all'과 'any' 사용
  - 'all'은 행에 있는 모든 값이 NaN일 때 해당 행을 삭제
  - 'any'는 하나의 NaN만 있어도 삭제
- dropna의 기본 설정은 'any'라서 모든 결측치를 지움

In [5]:
```
df_cleaned = df.dropna(how='all')
df_cleaned
```

Out [5]:

| | first_name | last_name | age | sex | preTestScore | postTestScore |
|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 |
| 2 | Tina | Ali | 36.0 | f | NaN | NaN |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 |

# 02 데이터 전처리의 전략

- 열 값이 모두 NaN일 경우에는 축(axis)을 추가하여 삭제

In [6]:
```
df['location'] = np.nan
df.dropna(axis=1, how='all')
```

Out [6]:

| | first_name | last_name | age | sex | preTestScore | postTestScore |
|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 |
| 1 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | Tina | Ali | 36.0 | f | NaN | NaN |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 |

- location이라는 열을 추가하여 값들을 모두 NaN으로 한 후 axis=1로 location 열만 삭제

# 02 데이터 전처리의 전략

- 매개변수 threshfh 데이터의 개수를 기준으로 삭제
  - thresh=1 지정하면 데이터가 한 개라도 존재하는 행은 남김
  - thresh=5 지정하면 데이터가 다섯 개 이상 있어야 남김

In [7]:
```
df.dropna(axis=0, thresh=1)
```

Out [7]:

| | first_name | last_name | age | sex | preTestScore | postTestScore | location |
|---|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 | NaN |
| 2 | Tina | Ali | 36.0 | f | NaN | NaN | NaN |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 | NaN |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 | NaN |

In [8]:
```
df.dropna(thresh=5)
```

Out [8]:

| | first_name | last_name | age | sex | preTestScore | postTestScore | location |
|---|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 | NaN |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 | NaN |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 | NaN |

# 02 데이터 전처리의 전략

## 1.2 채우기

- 채우기(fill) : 비어있는 값을 채움
- 일반적으로  드롭한 후에 남은 값들을 채우기 처리
- 평균, 최빈값 등 데이터의 분포를 고려해서 채움
- 함수 fillna 사용

In [9]:
```
df.fillna(0)
```

Out [9]:

| | first_name | last_name | age | sex | preTestScore | postTestScore | location |
|---|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 | 0.0 |
| 1 | 0 | 0 | 0.0 | 0 | 0.0 | 0.0 | 0.0 |
| 2 | Tina | Ali | 36.0 | f | 0.0 | 0.0 | 0.0 |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 | 0.0 |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 | 0.0 |

# 02 데이터 전처리의 전략

- 빈 값에 평균값을 채우려면 열 단위의 평균값을 계산하여 해당 열에만 값을 채움
  - 매개변수 inplace는 변경된 값을 리턴시키는 것이 아니고 해당 변수 자체의 값을 변경

In [10]:
```
df["preTestScore"].fillna(df["preTestScore"].mean(), inplace=True)
df
```

Out [10]:

| | first_name | last_name | age | sex | preTestScore | postTestScore | location |
|---|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 | NaN |
| 1 | NaN | NaN | NaN | NaN | 3.0 | NaN | NaN |
| 2 | Tina | Ali | 36.0 | f | 3.0 | NaN | NaN |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 | NaN |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 | NaN |

# 02 데이터 전처리의 전략

- 열별 분포를 고려하여 채울 수 있음
  - groupby 함수로 각 인덱스의 성별에 따라 빈칸을 채움

| | |
|---|---|
| In [11]: | df.groupby("sex")[<br>"postTestScore"].transform(<br>"mean") |
| Out [11]: | 0 43.5<br>1 NaN<br>2 70.0<br>3 43.5<br>4 70.0<br>Name: postTestScore, dtype: float64 |

# 02 데이터 전처리의 전략

- fillna 함수 안에 transform을 사용하여 인덱스를 기반으로 채울 수 있음
  - 일반적으로 쓰이는 기법

In [12]:

```
df["postTestScore"].fillna(
    df.groupby("sex")["postTestScore"].transform("mean"), inplace=True)
df
```

Out [12]:

| | first_name | last_name | age | sex | preTestScore | postTestScore | location |
|---|---|---|---|---|---|---|---|
| 0 | Jason | Miller | 42.0 | m | 4.0 | 25.0 | NaN |
| 1 | NaN | NaN | NaN | NaN | 3.0 | NaN | NaN |
| 2 | Tina | Ali | 36.0 | f | 3.0 | 70.0 | NaN |
| 3 | Jake | Milner | 24.0 | m | 2.0 | 62.0 | NaN |
| 4 | Amy | Cooze | 73.0 | f | 3.0 | 70.0 | NaN |